POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Gabriele Ginestroni**
Giacomo Gumiero
Lorenzo Iovine
Nicola Landini
Francesco Leone

Group Number: **10**

Academic Year: 2022-2023

Contents

1 | Introduction

In this chapter will be presented the problem specification and the hypothesis under which the database is implemented.

1.1. Problem Specification

This project aims to build a graph database that handles scientific articles contained in the DBLP bibliography. The focus is on creating a database which allows efficient information retrieval from the publications citation network. The main features analyzed are: articles, authors, venues, fields of study and the relations between all of them.

1.2. Assumptions

1. Data in the sources are heterogeneous and can be inconsistent
2. It is possible that an author writes for different organizations
3. Field *_id* in **author** exists and it is unique
4. Field *_id* in **article** exists and it is unique
5. It is possible to find a self-reference in a publication
6. Venues with *type=0* are Conference
7. Venues with *type=1* are Journal
8. There is no constraint between the publication type and the venue type in which it is published

2 | ER Diagram

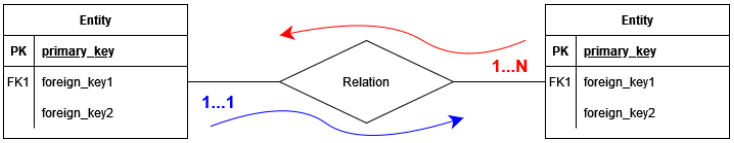


Figure 2.1: ER Diagram Organization

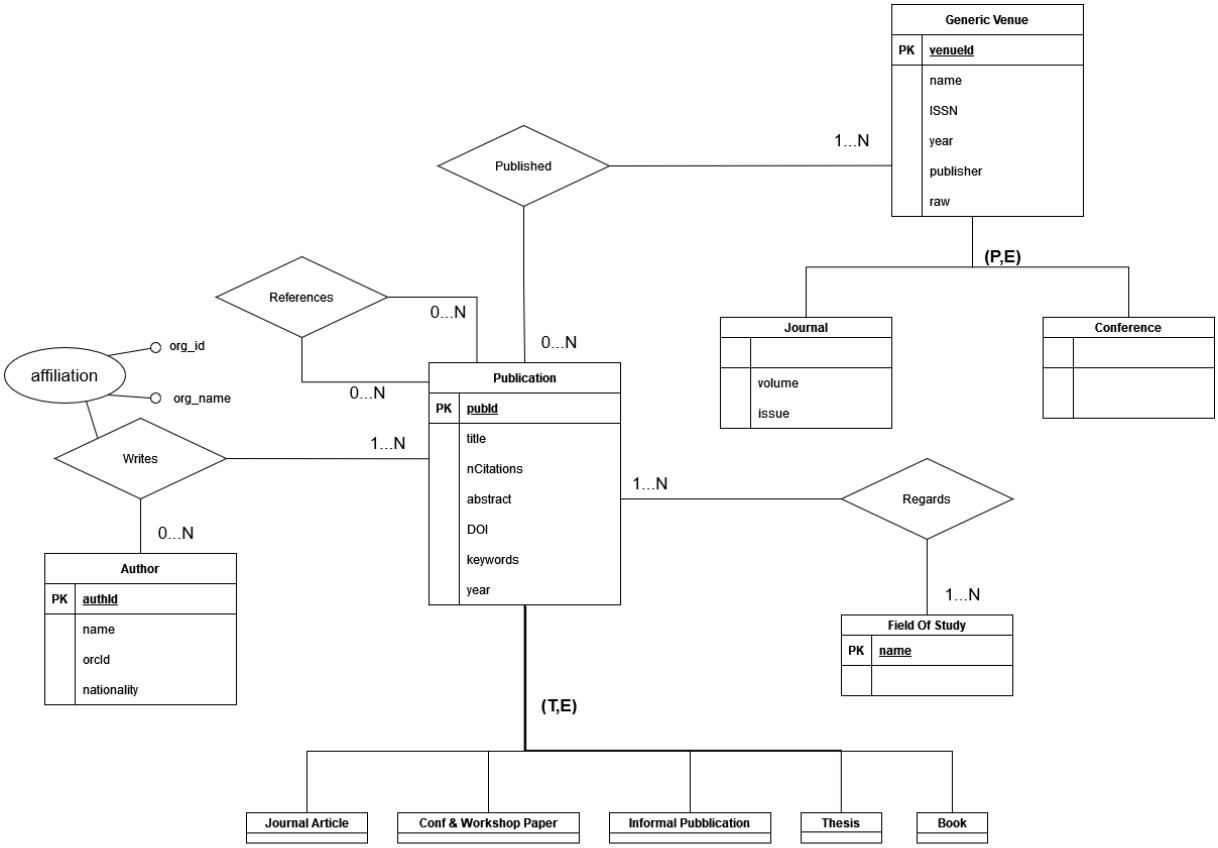


Figure 2.2: ER Diagram

The ER diagram designed contains the following entities:

- **Publication:** this entity represents all the scientific articles. They are identified by their key *pubId*. Other important attributes are: *title, nCitations, abstract, DOI, keywords, year*. Publication entity is the superclass of a Total and Exclusive ISA hierarchy with the following subclasses: *Journal article, Conference & Workshop paper, Informal publication, Thesis, Book*.
- **Author:** it is the one who contributed to a publication. Its key is *authId* and its attributes are: *name, orcid, nationality*.
- **Venue:** it is where a publication is published or presented. This is a superclass of a Partial and Exclusive ISA hierarchy with the two subclasses *Journal* and *Conference*. The key is *venueId* and its other attributes are: *name, ISSN, year, publisher, raw*.
- **Field Of Study:** this entity represents the subjects of the publication and its key is *name*.

The ER diagram designed contains the following relationships:

- **Writes:** is the relationship between *author* and *publication* which features the affiliation property characterized by org_id and the org_name. The affiliation is the institute where the author worked for the publication.
- **References:** is the relationship between a *publication* and another *publication* cited by the first.
- **Regards:** is the relationship between a *publication* and its *fields of study*.
- **Published:** is the relationship between a *publication* and its *venue*.

2.1. Important differences between ER and the implementation

- As it will be explained in the section 3.2.3, in the implementation we decided to distinguish between different venue nodes using the *raw* property. In this way we needed to move some venue attributes (volume, issue, publisher, issn, online_issn) to the relationship *Published*. This has been done due to the low data quality of the dataset with respect to the venue entity.
- The publication hierarchy has not been implemented because the dataset used doesn't distinguish between different types of publications. We decided to rep-

resent the conceptual ER in this way taking inspiration from the DBLP.org site description.

- Some attributes that have been included in the ER are missing in the implementation because either not relevant from queries point of view or dataset is missing those properties (e.g. *nationality* in *Author*).

3 | Dataset Description

3.1. Dataset Preprocessing

The dataset we used is based on DBLP-Citation-network V13 at `https://www.aminer.org/citation` whose size is 17GB. Using **Pandas Profiling** we analyzed the dataset focusing our attention on distinctness and completeness of the attributes. We used this result for selecting the property to be used during the merge operation (i.e. attributes with high values of distinctness and low missing values), to ignore fields that were not informative, and also to filter the tuples. Then we sliced the entire dataset obtaining a subset of 16MB. The slicing operation was not performed randomly, in order to avoid a subset of the dataset containing some publications but none of the publications referenced by them. The working sample of about 6400 publications was obtained in this way:

- first a partition of 20M rows was extracted from the original database so that we started from a pool of 257K articles instead of using all the original database of 5.4M articles, we used 5% as a partition to speed up the sample generation process
- then an extract of 4K publications was used as a base, we arbitrarily picked the ones from position 20000 to 24000 of the partition
- in the end the ids of those 4K publications were looked up inside our partition to obtain also articles that cited them, in order to have sufficient relationships to make different queries

The following is a part of the script used for the operation just described:

```
sed -n $start","$end"p" $input > tmp.json
while IFS='' read -r line || [[ -n "$line" ]]; do
  artId=$(echo "$line" | cut -d '"' -f4)
  grep "$artId" $input >> $output
done < tmp.json
```

We decide not to add any data to our dataset to preserve data coherency.

3.2. Attributes Description

In this section we will present all the attributes contained in the original dataset, pointing out which of them are considered or not.

3.2.1. Publication

Publication represent the central concept of the system and contains:

- **_id** is an alphanumeric string that was used during the merge operation, that's because is unique and every nodes has this property.
- **title** represents the title of the publication.
- **authors** is a set of authors.
- **venue** it is where a publication is published or presented.
- **year** represents the year of publication.
- **keywords** is a set containing keywords of the publication.
- **fos** is a set containing the fields of study of the publication.
- **n_citation** is the number of times that this publication has been mentioned.
- **page_start** defines the starting page of the publication. This attribute wasn't take into consideration because doesn't target the goal of the project.
- **page_end** defines the last page of the publication. This attribute wasn't take into consideration because doesn't target the goal of the project.
- **lang** represents the language of the publication.
- **volume** is the volume of the venue in which the article has been published. This is a property of the *Publication - Venue* relationship.
- **issue** refers to how many times a periodical has been published during that year.
- **issn** is an identification code of the venue of the publication. This is a property of the *Publication - Venue* relationship.
- **isbn** is an identification code of the venue of the publication. This attribute wasn't take into consideration.
- **doi** Digital Object Identifier is a persistent and standardized identifier. We didn't use this attribute as a key during the importing phase because a significant amount

of publication misses it.

- **pdf** contains a string that links to the publication PDF online. This attribute wasn't take into consideration due to the presence of many missing or null values.
- **url** contains a set of links to the publication resources online. This attribute wasn't take into consideration because doesn't target the goal of the project.
- **abstract** is a string containing a brief summary of the contents of the paper.
- **references** is a set of ids representing the publications mentioned.

3.2.2. Author

The dataset provides the following author fields:

- **_id** is an alphanumeric string that was used during the merge operation, that's because is unique and almost every nodes has this parameter.
- **name** is the name of the author.
- **org** is the name of the organization in which the author worked for a specific publication. It is an attribute of the relationship *Writes* described before
- **orgid** is the identifier of the organization in which the author worked for a specific publication. It is an attribute of the relationship *Writes* described before
- **gid** is an alternative to the orgid field. This attribute wasn't take into consideration due to the presence of many missing or null values.
- **orgs** is a set of organizations for which the author worked. This attribute wasn't take into consideration due to the presence of many missing or null values
- **email** is the email address of the author. This attribute wasn't take into consideration due to the presence of many missing or null values and because doesn't target the goal of the project
- **orcid** Open Researcher and Contributor ID is a unique identifier for authors of scientific articles. This attribute is taken into consideration although is not always present.
- **oid** is an identifier for the author. This attribute wasn't take into consideration due to the presence of many missing or null values.

- **bio** is a string that describes the author. This attribute wasn't take into consideration due to the presence of many missing or null values.
- **sid** is an identifier for the author. This attribute wasn't take into consideration due to the presence of many missing or null values.
- **name_zh** is a field that describe name in chinese language. This attribute wasn't take into consideration due to the presence of many missing or null values.
- **org_zh** is a field that describe the organization in chinese language. This attribute wasn't take into consideration due to the presence of many missing or null values.

3.2.3. Venue

Venue is where a publication is published or presented. It contains:

- **_id** is an alphanumeric identifier. We didn't use this property for the merge operation due to the large amount of missing values as previously stated.
- **raw** is the name or the abbreviation of the venue (regardless the year, issue or volume) in which the publication was presented. This property was used during the merge operation due to the low number of missing values and the fact that in this way publications were grouped based on the name or the abbreviation of the venue in which they were presented.
- **raw_zh** refers to the event (without specifying the year) in which the publication was presented. This attribute wasn't take into consideration due to the presence of many missing or null values.
- **type** indicates the type of the publication. Exploiting the profiling we understood the distribution of different values of *type*, and become clear that 0 and 1 were attributable respectively to conference and journal. The other values were not easily attributable to other types of publication, so we decided to consider as a generic venue:
  - every *type* values different from 0 or 1
  - every entity in which the *type* field is missing
- **sid** unclear and frequently missing attribute. This attribute wasn't take into consideration.

- **t** represents the type of the publication. This attribute wasn't take into consideration due to the presence of many missing or null values.
- **issn** is an identification code of the venue of the publication. We used the one in Publication instead of this one after seeing the profiling results.
- **name_d** is the extended name of the event or volume in which the publication was presented.
- **publisher** is the name of the publisher of the Venue.
- **online_issn** is an alternative identification code with respect to the issn. This is a property of the *Publication - Venue* relationship.

4 | Graph Diagram

The designed graph diagram presented on the next page represents the following nodes:

- **Publication**
- **Author**
- **Venue**
- **Field of Study**

The properties of the nodes are the ones described in the previous chapter.
The diagram also represents the following links:

- **Writes** links *Author* with the *Publication* that he wrote. His properties are *org* and *orgid* that represent the institute where the Author worked for that specific publication.
- **Published** links *Publication* with the *Venue* in which it was published. His properties are *issue*, *volume*, *issn*, *publisher* and *online_issn*.
- **Regards** links *Publication* with the *Fields of Study*.
- **References** links a *Publication* with another *Publication* it cites.

In the following diagram we decided to represent two *Authors*, two *Publications*, two *Fields of Study* and two *Venues* (one *Journal* and one *Conference*) in order to efficiently describe the system functionalities.
In this case *Author1* wrote both the *Publications*, while the other one wrote only *Publication2*. Both the *Publications* covered the same *Fields of Study* but *Publication1* was published in a conference, while the other one was published in a journal. Eventually a relationship *References* that links *Publication1* with *Publication2* exists meaning that *Publication1* cites *Publication2*.

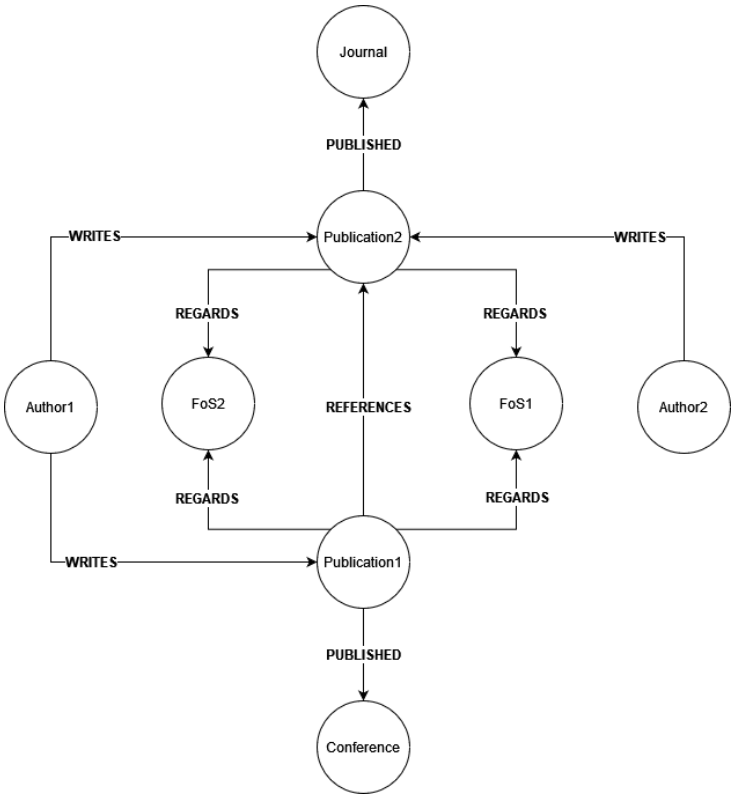


Figure 4.1: Graph Diagram

5 | Sample Dataset

In this chapter we will present the import commands that generates 39550 nodes distributed as follows:

- 6340 *Publication* nodes
- 17025 *Author* nodes
- 2133 *Conference* nodes
- 256 *Journal* nodes
- 506 *Generic Value* nodes
- 13210 *Field of Study* nodes

In order to complete these operations we used the plug-in apoc.

1. Create *Publication*, *Author* and `WRITES` relationship between them:

```
call apoc.load.json("test.json") yield value
UNWIND value AS pub
UNWIND pub.authors AS aut WITH aut,pub WHERE aut._id IS NOT NULL
MERGE (publication:Publication{id:pub._id}) ON CREATE SET
    publication.title  = pub.title,
    publication.doi = pub.doi,
    publication.year = pub.year,
    publication.n_citation = pub.n_citation,
    publication.keywords = pub.keywords,
    publication.abstract = pub.abstract,
    publication.lang = pub.lang
MERGE(author:Author{id:aut._id}) ON CREATE SET
    author.name = aut.name,
    author.orcid = aut.orcid
MERGE(author)-[writes:WRITES]->(publication) ON CREATE SET
    writes.org=aut.org,
    writes.orgid=aut.orgid
```

```
```

Added 23365 labels, created 23365 nodes, set 138863 properties, created 18534 relationships.

2. Create REFERENCE relationship between *Articles*:

```
call apoc.load.json("test.json") yield value
UNWIND value AS pub
UNWIND pub.references AS ref
MATCH (init:Publication{id:pub._id})
MATCH (final:Publication{id:ref})
MERGE(init)-[:REFERENCES]->(final)
```

Created 2866 relationships.

3. Create *Conference* when raw exists and type equal to 0:

```
call apoc.load.json("test.json") yield value
UNWIND value AS art
WITH art WHERE art.venue.raw IS NOT NULL AND art.venue.type = 0
MATCH(article:Publication{id: art._id})
MERGE(venue:Conference{raw:art.venue.raw}) ON CREATE SET
    venue.name = art.venue.name_d
MERGE(article)-[pub:PUBLISHED]->(venue) ON CREATE SET
    pub.issue = art.issue,
    pub.volume = art.volume,
    pub.issn = art.issn,
    pub.online_issn = art.venue.online_issn,
    pub.publisher = art.venue.publisher
```

Added 2133 labels, created 2133 nodes, set 22676 properties, created 4846 relationships.

4. Create *Journal* when raw exists and type equal to 1:

```
call apoc.load.json("test.json") yield value
UNWIND value AS art
WITH art WHERE art.venue.raw IS NOT NULL AND art.venue.type = 1
MATCH(article:Publication{id:art._id})
MERGE(venue:Journal{raw:art.venue.raw}) ON CREATE SET
    venue.name = art.venue.name_d
```

```
MERGE(article)-[pub:PUBLISHED]->(venue) ON CREATE SET
    pub.issue = art.issue,
    pub.volume = art.volume,
    pub.issn = art.issn,
    pub.online_issn = art.venue.online_issn,
    pub.publisher = art.venue.publisher
```

Added 256 labels, created 256 nodes, set 1681 properties, created 351 relationships.

5. Create *Generic Venue* when `raw` doesn't exist or `type` different from 0 or 1:

```
call apoc.load.json("test.json") yield value
UNWIND value AS art
WITH art WHERE art.venue.raw IS NOT NULL AND (art.venue.type IS NULL OR (
   art.venue.type <> 1 and art.venue.type <> 0))
MATCH(article:Publication{id:art._id})
MERGE(venue:GenericVenue{raw:art.venue.raw}) ON CREATE SET
    venue.name = art.venue.name_d
MERGE(article)-[pub:PUBLISHED]->(venue) ON CREATE SET
    pub.issue = art.issue,
    pub.volume = art.volume,
    pub.issn = art.issn,
    pub.online_issn = art.venue.online_issn,
    pub.publisher = art.venue.publisher
```

Added 586 labels, created 586 nodes, set 5122 properties, created 1116 relationships.

6. Create *Field of Study* and `REGARDS` relationship between *FoS* and *Publication*:

```
call apoc.load.json("test.json") yield value
UNWIND value AS pub
UNWIND pub.fos AS f
MATCH(publication:Publication{id:pub._id})
MERGE(fos:Fos{name:f})
MERGE (publication)-[:REGARDS]->(fos)
```

Added 13210 labels, created 13210 nodes, set 13210 properties, created 59740 relationships.

6 | Commands and Queries

6.1. Commands

We have identified the following `INSERT` and `UPDATE` commands to show the system basic functionalities.

6.1.1. Insert two authors in the system

Assuming they are not present in the dataset, we used the `CREATE` to create two instances of *Author*.

```
CREATE (author1:Author {id: "54857748dabfae8a11fb2a1e", name: "Emanuele Della
    Valle", orcid: "0000-0002-5176-5885"})
CREATE (author2:Author {id: "53f487fbdabfaee4dc8b1e68", name: "Alessio Bernardo
    ", orcid: "0000-0002-3492-0345"})
```

6.1.2. Insert Journal Venue where the paper has been published

Assuming they are not present in the dataset, we used the `CREATE` to create an instance of *Venue (Journal)* specifying where it was published.

```
CREATE (journal:Journal {raw: "ESA", name: "Expert Systems with Applications"})
```

6.1.3. Create (if do not exist) publication's fields of study

We used the `MERGE` to create three instances of *Field Of Study*

```
MERGE (fos1:Fos{name:"Computer Science"})
MERGE (fos2:Fos{name:"Stream Reasoning"})
MERGE (fos3:Fos{name:"Big Data"})
```

6.1.4. Create a publication and his relationships with already existing entities

At the beginning we MATCH two specific *Authors*, three specific *Fields of Study* and one specific *Journal*. Then, we create the *Publication* and the relationships with the entities listed before

```
MATCH (author1:Author),(author2:Author) WHERE author1.id = "54857748
    dabfae8a11fb2a1e" AND author2.id = "53f487fbdabfaee4dc8b1e68"
MATCH (fos1:Fos),(fos2:Fos),(fos3:Fos) WHERE fos1.name="Computer Science" AND
    fos2.name="Stream Reasoning" AND fos3.name="Big Data"
MATCH (journal:Journal) WHERE journal.raw = "ESA"
CREATE (pub:Publication{id: "53e99f86b7612d9702859fdf",
    doi: "10.1016/j.eswa.2022.116630",
    title: "An extensive study of C-SMOTE, a Continuous Synthetic Minority
    Oversampling Technique for Evolving Data Streams",
    year: 2022,
    n_citation: 3,
    keywords: ["Evolving Data Stream","Streaming","Concept drift","Balancing"],
    abstract: "Streaming Machine Learning (SML) studies algorithms that update
    their models, given an unbounded and often non-stationary flow of data
    performing a single pass.",
    lang: "en"})

//Creation of relationship WRITES between authors and the article
CREATE (author1)-[:WRITES{org: "Politecnico di Milano",
    orgid: "5b86c975e1cd8e14a3d351a3"}]->(pub),
    (author2)-[:WRITES{org: "Politecnico di Milano",
    orgid: "5b86c975e1cd8e14a3d351a3"}]->(pub)

//Creation of relationship REGARDS between article and fields of study
CREATE (pub)-[:REGARDS]->(fos1),
    (pub)-[:REGARDS]->(fos2),
    (pub)-[:REGARDS]->(fos3)

//Creation of relationship PUBLISHED between the article and the journal
//Note that in this case fields online_issn and issue are not available
CREATE (pub)-[published:PUBLISHED{volume:"196",
    issn: "0957-4174",
    publisher:"Elsevier"}]->(journal)
```

6.1.5. Create REFERENCES relationship among the article and the cited ones

We MATCH the two publications referred by our article and we create REFERENCES relationships that link the publications already present in the dataset with the article just created

```
MATCH (p1:Publication),(p2:Publication),(p:Publication{id:'53
    e99f86b7612d9702859fdf'})
WHERE p1.id = "53e99fe4b7602d97028bf743" AND p2.id="53e99fddb7602d97028bc085"
CREATE (p)-[:REFERENCES]->(p1), (p)-[:REFERENCES]->(p2)
```

6.1.6. Increment cited publications' n_citation attribute by 1

It's an update command that increments the number of citations of the two articles cited by the article that we created in the previous commands

```
MATCH (p1:Publication), (p2:Publication)
WHERE p1.id = "53e99fe4b7602d97028bf743" AND p2.id="53e99fddb7602d97028bc085"
SET p1.n_citation = p1.n_citation + 1, p2.n_citation = p2.n_citation + 1
```

6.2. Queries

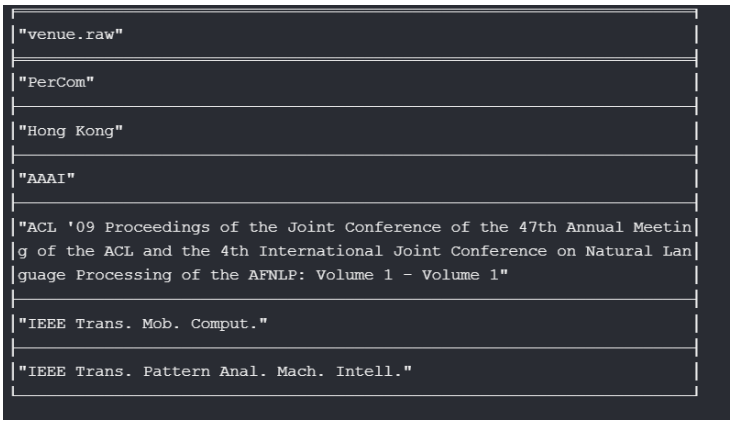
We have identified the following queries in order to show the system's basic functionalities.

6.2.1. Query 1

This query returns all the venues where have been published the articles of one of the authors that have written the max number of articles

Description: for each author we count how many articles he wrote, then we order by the count of written articles by descending order and keep only the top 1 author. Eventually we match all the articles written by this author and we return the raw of the venues in which these articles were published

```
MATCH(author:Author)-[:WRITES]->(article:Publication)
WITH author, count(*) AS articleWritten
ORDER BY articleWritten DESC
WITH author AS authorMax, articleWritten LIMIT 1
MATCH(authorMax:Author)-[:WRITES]->(article)
MATCH(article:Publication)-[:PUBLISHED]->(venue)
RETURN DISTINCT venue.raw
```



"venue.raw"
"PerCom"
"Hong Kong"
"AAAI"
"ACL '09 Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1"
"IEEE Trans. Mob. Comput."
"IEEE Trans. Pattern Anal. Mach. Intell."

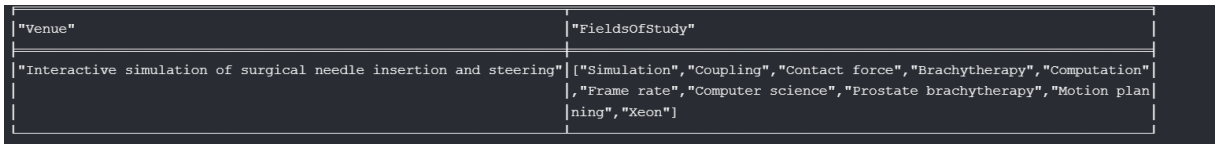
Figure 6.1: Results: 6 records

6.2.2. Query 2

Starting from the author found in query 1 we want to find the article with the max number of co-authors and return the venue and fields of study

Description: we match a specific author given its id, then we match all his articles and then all the co-authors of his articles. Then, for each article we count the number of co-authors and order them by descending order w.r.t. the number of co-authors. We keep the top article, then we match its venues and its fields of study and return them by collecting the fos into a list

```
MATCH(author:Author{id:'548d281cdabfae8a11fb4ea1'})
MATCH(author)-[:WRITES]->(article)
MATCH(article)<-[:WRITES]-(coAuth)
WITH article , count(*) AS nCount
ORDER BY nCount DESC LIMIT 1
MATCH(article)-[:PUBLISHED]->(venues)
MATCH(article)-[:REGARDS]->(fos)
RETURN venues.raw AS Venue, collect(fos.name) AS FieldsOfStudy
```



"Venue"	"FieldsOfStudy"
"Interactive simulation of surgical needle insertion and steering"	["Simulation","Coupling","Contact force","Brachytherapy","Computation","Frame rate","Computer science","Prostate brachytherapy","Motion planning","Xeon"]

Figure 6.2: Result: 1 record

6.2.3. Query 3

This query finds all authors that have worked together more than once on the same field of study

Description: we match 2 different authors and 2 different articles in which they collaborated, matching also all their fields of study, then we filter by keeping only those who have at least one field of study in common. Eventually we return the name of the two authors

Note: the condition on the authors id needed to ensure that every couple is returned only once

Note: query result picture is partial

```
MATCH(aut1:Author)-[:WRITES]->(art1)-[:REGARDS]->(fos1)
MATCH(aut1:Author)-[:WRITES]->(art2)-[:REGARDS]->(fos2)
MATCH(aut2:Author)-[:WRITES]->(art1)
MATCH(aut2:Author)-[:WRITES]->(art2)
WHERE (art1)<-[:WRITES]-(aut2) AND
    (art2)<-[:WRITES]-(aut2) AND
    art1.id <> art2.id AND aut1.id > aut2.id
    AND fos1 = fos2
RETURN DISTINCT aut2.name, aut1.name
```

"aut2.name"	"aut1.name"
"Shinya Kiriyama"	"Keikichi Hirose"
"Simon X. Yang"	"Fangju Wang"
"Siva G. Narendra"	"Volkan Kursun"
"Eby G. Friedman"	"Volkan Kursun"
"Eby G. Friedman"	"Siva G. Narendra"
"Siva G. Narendra"	"Vivek K. De"
"Eby G. Friedman"	"Vivek K. De"

Figure 6.3: Results: 526 records

6.2.4. Query 4

Find shortest path of WRITES links between an author that wrote a publication for a Stanford University and another author that wrote an article for a California University.

Description: we match 2 authors such that one has published at least once in affiliation with Stanford university and the other with California university. We check that the 2

authors are not the same and then we compute the shortest path between them (composed of WRITES relationships) bounded to 5 steps. Then for all the obtained paths we keep the ones with length > 2 to avoid trivial path.

Note: graph image is a subset of actual query result (PATH: Kunle ->...-> Krste)

```
MATCH (auth1:Author), (auth2:Author)
WHERE EXISTS {
    MATCH (auth1)-[w:WRITES]->()
    WHERE w.org CONTAINS 'Stanford'}
AND EXISTS {
    MATCH (auth2)-[v:WRITES]->()
    WHERE v.org CONTAINS 'California'}
AND auth1 <> auth2
MATCH p = shortestPath((auth1)-[:WRITES*1..5]-(auth2))
WHERE length(p) > 2
RETURN p, auth1.name, auth2.name
```

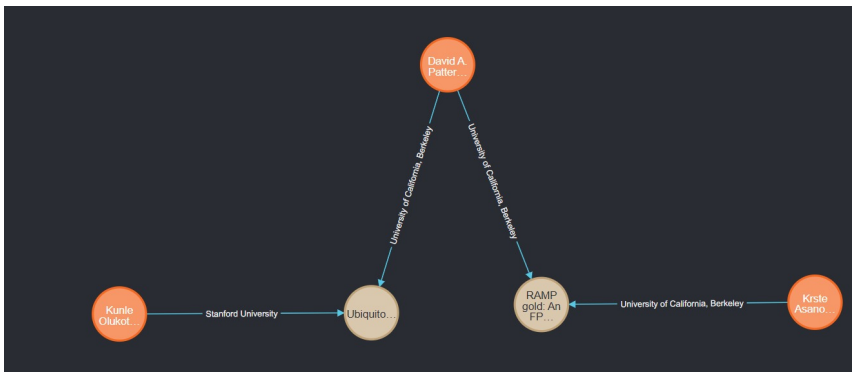


Figure 6.4: Results: 18 records, only 1 path in the figure for readability

6.2.5. Query 5

Find the venue with the highest average number of citations of its publications starting from year 1990 and return the most frequent field of study of its publications.

Description: we match all the publications that have the year attribute > 1990, then we group with respect to the venue they've been published. Then for each venue we compute the average of the number of citations of the articles that were published on it. After that we order the averages by descending order and keep the greatest one. Then we match all the fields of study of all the articles (ignoring the publication date) that have been published on that venue. After that, we count those fields of study, order by descending order and keep only the most frequent one. Finally we return the venue, its most frequent

field of study and the number of occurrences of that field of study

```
MATCH (v)<-[p:PUBLISHED]-(a:Publication)
WHERE a.year > 1990 AND a.n_citation IS NOT NULL
WITH v, avg(a.n_citation) AS mean, count(a) AS n_article
ORDER BY mean DESC
WHERE n_article > 5
WITH v AS venue, mean, n_article LIMIT 1
MATCH (fos)<-[:REGARDS]-(art)-[:PUBLISHED]->(venue)
WITH fos, count(*) AS fosN, venue
ORDER BY fosN DESC LIMIT 1
RETURN venue.raw AS VenueRaw, fos.name AS FieldOfStudy, fosN AS
    FieldOfStudyOccurrence
```

"VenueRaw"	"FieldOfStudy"	"FieldOfStudyOccurrence"
"International Journal of Computer Vision"	"Computer vision"	6

Figure 6.5: Result: 1 record

6.2.6. Query 6

Find the 3 most common fos (and their count) of the articles published on the volume of *Clinical Orthopaedics and Related Research* that received the greatest number of total citations.

Description: we match the specific journal using the raw, then we aggregate with respect to the volume in which articles were published and sum their number of citations. Then we order by descending order and keep the 1st volume. Finally, we match all the relationships with that volume involving the same venue as before, grouping by fields of study and counting their occurrences. We return the top 3 fields of study according to their count.

```
MATCH (j:Journal{raw:'Clinical Orthopaedics and Related Research'})<-[p:
    PUBLISHED]-(art)
WITH p.volume as pVolume, sum(art.n_citation) as totalVolumeCitations
ORDER BY totalVolumeCitations DESC LIMIT 1
MATCH (j:Journal{raw:'Clinical Orthopaedics and Related Research'})<-[p:
    PUBLISHED{volume:pVolume}]-(art)
MATCH (art)-[:REGARDS]->(fos)
WITH fos, count(*) as fosCount
ORDER BY fosCount DESC LIMIT 3
RETURN fos.name as fieldOfStudy,fosCount
```

"fieldOfStudy"	"fosCount"
"Wireless"	1
"Broadcasting"	1
"Computer science"	1

Figure 6.6: Results: 3 records

6.2.7. Query 7

Find the authors of the article that cites articles with the greatest number of different years of publication.

Description: we match in a subquery the REFERENCES relationships and return all the article-year(of the referenced article) pairs. Then, we count for each article all the year values returned by the previous subquery. We order the articles according to the count in descending order and keep the top one. Then we match the authors of the top 1 article and return their names.

```
CALL {
    MATCH (art1:Publication)-[:REFERENCES]->(art2:Publication)
    RETURN DISTINCT art1 , art2.year AS year
}
WITH art1, count(year) as nCount
ORDER BY nCount DESC LIMIT 1
MATCH (author)-[:WRITES]->(art1)
RETURN COLLECT(author.name) AS authors, nCount
```

"authors"	"nCount"
["Miro Kraetzl","Jop F. Sibeyn","Miltos D. Grammatikakis","D. Frank Hsu"]	6

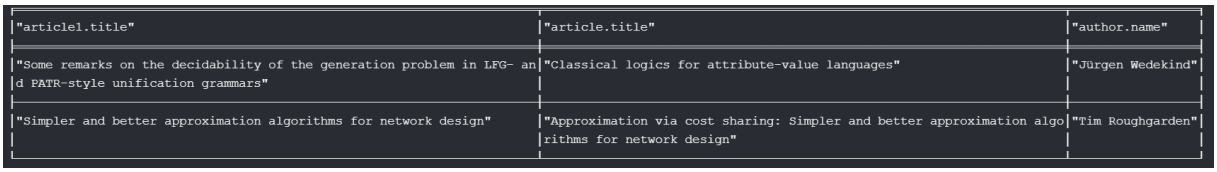
Figure 6.7: Results: 1 record

6.2.8. Query 8

Find the authors that have written an article for *Stanford* and an article for another organization.

Description: We match for each author 2 publications such that one has been written in affiliation with *Stanford* and the other one not (the previous condition ensures that the two articles are different). Finally we return the articles' title and their author name.

```
MATCH (article)<-[w1:WRITES]-(author)-[w2:WRITES]->(article1)
WHERE w1.org CONTAINS "Stanford" AND NOT w2.org CONTAINS "Stanford"
RETURN article1.title,article.title, author.name
```



"article1.title"	"article.title"	"author.name"
"Some remarks on the decidability of the generation problem in LFG- and PATR-style unification grammars"	"Classical logics for attribute-value languages"	"Jürgen Wedekind"
"Simpler and better approximation algorithms for network design"	"Approximation via cost sharing: Simpler and better approximation algorithms for network design"	"Tim Roughgarden"

Figure 6.8: Results: 2 records

6.2.9. Query 9

Find the shortest path (of length of at least 2) of references between a specific publication and a publication written in affiliation with *Stanford University*.

Description: we match the publication with a specific id and another publication that has been written in affiliation with *Stanford*. We compute the shortest path of REFERENCES relationships connecting them, bounding the search to 5 steps of depth).

```
MATCH (a:Publication{id:'53e9a0eeb7602d97029d96d3'}),
    (a2:Publication),
    p = shortestPath((a)-[:REFERENCES*1..5]->(a2))
WHERE EXISTS {
    (auth1)-[w:WRITES]->(a2)
    WHERE w.org CONTAINS 'Stanford'
}
AND length(p) > 2
RETURN p
```

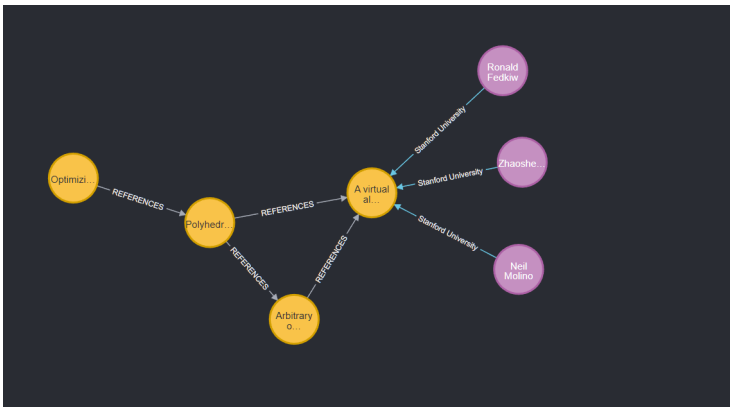


Figure 6.9: Result: 1 record

6.2.10. Query 10

All articles with venue SIGIR written by authors that combined have all 26 letters of the alphabet.

Description: We match all the articles published on a specific venue, then we match all the writes relationships and for each publication we combine all the authors' name letters together and keep the publications where the combined authors' letters are covering the entire alphabet (case insensitive and ignoring special characters). Finally we return the publication title and the list of all the authors' names

```
MATCH (art:Publication)-[:PUBLISHED]->(venue{raw: 'SIGIR'})
MATCH (art)<-[w:WRITES]-(aut:Author)
WITH art, apoc.text.join(COLLECT(aut.name), ', ') AS authors
WITH art, authors, split(toLower(authors), "") AS letterList
UNWIND letterList AS letters
WITH DISTINCT art, authors, letters
WHERE letters =~ '[a-z]'
WITH art, authors, COUNT(letters) AS diffLetters
WHERE diffLetters = 26
RETURN art.title AS title, authors
```

"title"	"authors"
"Building an information retrieval test collection for spontaneous conversational speech"	"James Mayfield, Liliya Kharevych, Dagobert Soergel, Martin Franz, David S. Doermann, Douglas W. Oard, Samuel Gustman, Bhuvana Ramabhadran, Stephanie Strassel, Jianqiang Wang, G. Craig Murray, Xiaoli Huang"

Figure 6.10: Result: 1 record

6.2.11. Query 11

Find the article with most distinct university affiliations.

Description: We match all the `WRITES` relationships (we could have used the `WITH` keyword instead of `CALL`). This subquery returns all the distinct article-affiliation pairs. Then we group w.r.t. the article and count its affiliations. We order the articles by counting in descending order and keep only the one with the greatest number of different affiliations. **Note:** this query complexity is of 2 nodes, but we decided to present it because of the relevance of the results.

```
CALL{
    MATCH(article:Publication)<-[w:WRITES]-(author:Author)
    RETURN DISTINCT article, w.org as org
}
WITH  article, count(*) as n
ORDER BY n DESC LIMIT 1
RETURN  article.title,  n as AffiliationsCount
```

"article.title"	"AffiliationsCount"
"Cluster Analysis and Decision Trees of MR Imaging in Patients Sufferi ng Alzheimer's"	12

Figure 6.11: Result: 1 record

7 | Conclusion

Some interesting conclusions can be drawn from the development of this project: designing a Graph DB allows us to open a new perspective in database creation. This technology enables an efficient visualization of the design choices and it is more flexible than classical relational databases.
Other very useful aspects of this project were the dataset pre-processing operation and the profiling that allows us to deal with a real world problem of managing data.